Survey of Virtual Machine Research -- R. P. Goldberg, 1974

Short Summary: Virtual Machines are useful for running multiple OSes on the same H/W, software reliability, data security etc., However, the disadvantage is that the performance overhead is high and this could be the reason for their limited deployment until now.

--

Software simulation is used for testing and developing software for the architecture of a machine in development. However, this results in as much as 1000 times slowdown. Simulation of a machine architecture on the same machine improves performance, and is useful for providing a separate copy of the machine to each of the users. The simulated machines are called Virtual Machines (VM) and the simulator is called Virtual Machine Monitor(VMM).

Multi-programming OSes (priveleged software nuclei) provide an extended machine interface which is different from the bare machine interface (priveleged instructions are not exposed to the programs). Limitations of this model: multiple OSes cannot run on a single machine causing problems with transportability of user programs written for another OS; OS development, and running of testing and diagnostic programs becomes difficult.

Solution: VMs. VMM is the software which provides the illusion of many virtual machines on top of a single real machine. VM Example:IBM VM/370

To achieve good performance, all of the code which executes in non-priveleged mode is executed directly on the real machine. The priveleged mode is simulated in software by the VMM using a software virtual mode bit. To take care of the paging in VMs, the VMM maintains the page tables of the VMs and manipulates the page tables on the real machine to provide a notion of virtual storage to the VMs (Is this of similar complexity as the maintaining of per-process page tables in current OSes?). I/O instructions are priveleged and they trap to the VMM, which performs the I/O on behalf of the VM. Benefit: map one virtual I/O device to a different real device, or provide a virtual device for which there is no real counterpart. I/O is a major performance hit and the assumption is that I/O is of ``relatively low frequency.''

To improve the performance of VM systems, virtualizable architectures (Hardware Virtualizers) have been proposed. These architectures provide support for defining a mapping between the virtual and real machines called f-map. The f-map is invisible to all software and is controlled by the VMM. f-map violations cause a VM-fault to the VMM. This also provides for a recursive VM model (using composed f-maps), with faults passed to the VMM at the appropriate level. Example:IBM VM-assist for VM/370. The claim is that if designed properly, these hardware virtualizers perform as well as the real machine, due to program locality.

Thus, sources of overhead in VM systems include maintaining the status of the virtual processor, supporting priveleged instructions, paging in VMs, I/O instructions, and console functions (simulation of operator panel and lights in software :-)

Other performance improving strategies are policy approaches (assigning priorities to VMs, `increasing' VM memory to decrease the file I/O, and streamlining the VM by not using some instructions), compromising the VM interface (impure VMs) by moving some of the OS functions into the VM (disadvantage: programs running on this VM might be incompatible with the real machine), and improved mechanisms such as VM-assist.

A performance problem unique to VMs is that the resource allocation algorithms in the OS running above the VM might conflict with the ones in the VMM itself.

Other advantages of VMs: VMs help solve the "upgrade problem", by allowing users to run both the versions of the system in different VMs. Also, if a new device is developed, it can be easily integrated into the system (without modifying the OSes running on the machine) by mapping the new device to a virtual device already known to the OSes. Though this needs modification to the VMM, the assumption is that the VMM is simpler to modify and is the only software needing modification. VMs can be used for debugging of network software, and to provide software reliability (assumption:VMM is small and easy to verify) and data security.

Aside: The performance of VMware is pretty good compared to that given in the paper. I tried comparing VMware with native linux by compiling Linux kernel on it. The overhead seemed to be only 30%. Note that compilation is a I/O bound process, and it is supposed to cause the worst performance in VM systems.

Question: In the past, hardware was costly, and software was cheaper. This made simulation of multiple machines in software useful. However, today the situation is almost the reverse. Are VM systems applicable in today's scenario? Are applications such as software reliability and data security compelling enough to necessitate VM systems?